We claim:

1. A file system for a client computer system which comprises main memory and at least one secondary storage device, where said file system is programmed to receive and service file requests, to control accesses (including reads and writes) to a main memory, to group files together in clusters, and to store and retrieve clusters from said at least one secondary storage device, and where said file system comprises file system clustering logic which assists in said grouping of files together in clusters by grouping together files likely to be requested from said file system in close temporal proximity.

2. The file system of claim 1, further comprising:

a library of functions provided to applications by said file system, a function which takes as arguments at least two file names, which, when called, indicates to said file system clustering logic that said at least two file names provided as arguments should be stored together in one cluster if possible.

3. The file system of claim 1, where said file system clustering logic examines historical calls to files in order to determine which files are likely to be requested from said file system in tandem or close temporal proximity.

4. The file system of claim 3, further comprising, as part of a library of functions provided to applications by said file system, a collocation function which takes as arguments at

least two file names, which, when called, indicates to said file system clustering logic that the files named by the file names provided as arguments should be stored together in one cluster if possible.

5. A caching proxy system comprising a computer system utilizing the file system of claim 4, and programmed to receive and serve requests for data from a large distributed-data network.

6. The caching proxy system of claim 5, where said computer system is programmed to utilize said collocation function to provide an indication to said file system that the files named by the file names provided as arguments should be stored together in one cluster if possible.

7. The caching proxy system of claim 5, where said large distributed-data network is the World Wide Web.

8. The file system of claim 1, where said file system is further programmed to maintain and update meta-data, including file system meta-data stored in said main memory regarding said file system's usage of said main memory; file meta-data information regarding files stored in said main memory which have not yet been grouped into a cluster; and cluster meta-data regarding clusters stored in one of said at least one secondary storage device.

9. The file system of claim 8, where said file meta-data comprises a hash table and file information including a file number, file location in main memory or in a cluster, the status of the file, and a reference count of the number of users currently reading the file.

10. The file system of claim 9, where said maintenance and updating of file meta-data comprises the steps of computing a hash for the file name of each file being accessed and updating the file meta-data for the file at the corresponding entry in said hash table, or, if the corresponding entry does not match said file name, updating the file meta-data for the next consecutive entry in said hash table.

11. The file system of claim 8, where file system is further programmed to, upon loss of meta-data, scan said at least one secondary storage device after recovery and rebuild said meta-data.

12. The file system of claim 8, where file system is further programmed to record said meta-data in said at least one secondary storage device, and, upon loss of meta-data, to recover said meta-data from said at least one secondary storage device.

13. The file system of claim 8, where said file system is further programmed to access said file meta-data to determine the location of a file being requested.

14. The file system of claim 8, where said cluster meta-data information comprises for each cluster stored on said at least one secondary storage device, information about the *status* (empty, on disk, in memory, uncacheable) of the cluster, information about the last time the cluster was accessed, and a linked list of files in the cluster.

15. The file system of claim 14, where the size of said clusters does not vary.

16. The file system of claim 14, where the size of said clusters varies.

17. The file system of claim 16, where the size of said clusters is selected from a finite subset of possible cluster sizes.

18. The file system of claim 14, where the size of said clusters is selected using information about the size of files stored together in clusters.

19. The file system of claim 14, where the size of said clusters are based on powers of two.

20. The file system of claim 1, further comprising a library of functions provided to applications by said file system, comprising a write function which, when called, writes a given file directly to said at least one secondary storage device.

21. The file system of claim 20, where said write function, when called, removes the given file from memory.

22. The file system of claim 20, where said writing of a given file directly to said at least one secondary storage device is delayed until more space is needed in said main memory.

23. The file system of claim 20, where said writing of a given file directly to said at least one secondary storage device is delayed until said file system is idle.

24. The file system of claim 20, where said writing of a given file directly to said at least one secondary storage device is delayed until either more space is needed in said main memory or until said file system is idle.

25. The file system of claim 1, further comprising a library of functions provided to applications by said file system, comprising a read function which, when called with the name of a requested file, reads a cluster containing said requested file to said main memory if said requested file was not available in said main memory, and, upon completion of said reading of said cluster containing the requested file to said main memory or if said requested file was available in said main memory, returns a pointer to said requested file in main memory and protects said requested file in said main memory.

26. The file system of claim 25, said library of functions further comprising:

a done read function which, when called with a pointer returned by said read function call, indicates that said protection of said requested file in main memory prompted by said read function call should end.

27. The file system of claim 1, where said file system further comprises a daemon which groups files together in clusters and stores clusters to said at least one secondary storage device.

28. The file system of claim 27, where the operations of said daemon occur when more space is needed in said main memory.

29. The file system of claim 27, where the operations of said daemon occur when said file system is idle.

30. The file system of claim 27, where the operations of said daemon occur when more space is needed in said main memory or when said file system is idle.

31. The file system of claim 1, where said file system accesses file system parameter data, comprising data on file and cluster lifetimes.

32. The file system of claim 31, where said file system further comprises a disk space daemon which uses said file system parameter data in order to clean said at least one secondary storage device.

33. The file system of claim 31, where said file system further comprises a pack files daemon which performs said grouping of files together in clusters.

34. The file system of claim 33, where said pack files daemon packs the least recently used files into clusters.

35. The file system of claim 33, where said pack files daemon packs the most recently used files into clusters.

36. The file system of claim 33, where said file system further comprises at least two of said pack files daemons which perform said grouping of files together in clusters, at least one of which packs the least recently used files into clusters, and at least one other of which packs the most recently used files into clusters.

37. The file system of claim 31, where said file system, during operation, can change and reset said data on file and cluster lifetimes.

38. The file system of claim 37, where said changing and resetting of said data on file and cluster lifetimes uses access information about accesses of said file system.

39. The file system of claim 1, where said file system accesses file system parameter data, comprising at least one of: file lifetime, cluster lifetime, cluster size, cluster threshold, memory cache threshold to begin cleaning, memory cache eviction policy, file hash table size, disk data layout policy, delete file log threshold, size of delete log, number of hot clusters, hot cluster threshold and the size and name of the said at least one secondary storage device.

40. The file system of claim 1, where said file system is further programmed to identify clusters stored in said main memory as hot clusters, and to store said hot clusters on a prespecified area of one of said at least one secondary storage device.

41. The file system of claim 40, where said file system is further programmed to recover from a crash or other abnormal functioning by reading said hot clusters from said prespecified area of one of said at least one secondary storage device and writing said hot clusters to said main memory.

42. The file system of claim 1, where said file system is further programmed to identify clusters stored in said main memory as hot clusters, and to store the locations information regarding said hot clusters on a prespecified area of one of said at least one secondary storage device.

43. The file system of claim 40, where said file system includes a hot cluster daemon which performs said identification and storage of hot clusters.

44. The file system of claim 43, where said hot cluster daemon runs during system idle time.

45. The file system of claim 43, where said hot cluster daemon runs when a hot cluster threshold is reached.

46. The file system of claim 43, where said hot cluster daemon runs when called.

47. The file system of claim 43, where said hot cluster daemon runs during system idle time, when a hot cluster threshold is reached, or when called.

48. The file system of claim 1, where said file system is further programmed to log deletions of files and clusters in a delete log which is stored on one of said at least one secondary storage device.

49. The file system of claim 48, where said logging of deletions occurs during system idle time.

50. The file system of claim 48, where said logging of deletions occurs when a threshold on delete log size is reached.

51. The file system of claim 48, where said logging of deletions occurs when requested.

52. The file system of claim 48, where said logging of deletions occurs when logging has not occurred for a specified period of time.

53. The file system of claim 48, where said logging of deletions occurs during system idle time, when a threshold on delete log size is reached, when requested, or when logging has not occurred for a specified period of time.

54. The file system of claim 48, where said delete log is used to recover from a crash or other abnormal functioning.

55. The file system of claim 48, where said file system accesses file system parameter data, including delete file log threshold data and delete log size data, where possible size of said delete log is given by delete log size data, and where said file system is further programmed to monitor said delete log, and to clean said delete log when said delete log reaches the size given by said delete file log threshold data.

56. The file system of claim 1, where said file system is further programmed to precompute checksums for a transmission protocol for stored files and to send the precomputed checksum when the files are accessed.

57. The file system of claim 56, where said transmission protocol is the Transmission Control Protocol (TCP).